ARMY RESEARCH LABORATORY

Preliminary Study of a Hybrid Genetic Algorithm/Expert System for Modeling Complex Radar Signatures

Geoffrey H. Goldman

ARL-TR-2028

October 1999

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

Army Research Laboratory

Adelphi, MD 20783-1197

ARL-TR-2028 **October 1999**

Preliminary Study of a Hybrid Genetic Algorithm/Expert System for Modeling Complex Radar Signatures

Geoffrey H. Goldman

Sensors and Electron Devices Directorate

Abstract

I made an initial study of a hybrid genetic algorithm/expert system (HGAES) to model targets with nonlinear radar imaging effects caused by features such as cavities and canopies. The model for the nonlinear parameters was relatively simple, so it should be suitable for incorporation into hardware-in-the-loop and software-in-the-loop simulations that currently use point scatter models. I demonstrated the algorithm on simulated two-dimensional (2-D) inverse synthetic aperture radar (ISAR) images using a simple technique to determine the initial scattering centers. Many of the ideas used in developing the algorithm can be extended to more complex targets and from 2-D to 3-D images. A major issue in the development of an HGAES is knowledge representation. My conclusions are that models determined using this technique have the potential to enhance the accuracy of weapon systems simulations; thus, this technique is worth further investigation.

Contents

Figures

Introduction

Software-in-the-loop (SIL) and hardware-in-the-loop (HWIL) simulations provide a low-cost alternative to flight-testing weapon systems [1]. These techniques require computationally efficient target models that generate statistically accurate radar target signatures [2–4]. Simulated radar signatures can be efficiently computed using isotropic point scatter models. The radar cross section (RCS) and the position of point scatterers can be calculated from measurements rather than physics. This often leads to statistically accurate results but an inaccurate representation of the phenomenology, such as traveling and creeping waves, cavity resonance, and multibounce. Even with the most sophisticated electromagnetic (EM) simulation codes, the radar signatures of complex targets are difficult to accurately model. As more sophisticated signal processing algorithms are being developed, a higher degree of accuracy is required to model the target signatures. One approach to increasing model accuracy without adding numerous computational requirements is to enhance point scatter models.

For complex targets, point scatter models are typically generated from measured inverse synthetic aperture radar (ISAR) images. Figures 1 and 2 are a photograph and a two-dimensional (2-D) ISAR image measured at X-band of a Hind-D helicopter with aluminum foil covering the windows and engine inlets.

The brightness at the top of the image in figure 2 is probably due to radiation resonating in various cavities in the target where it is delayed in time and dispersed in frequency. It is difficult to capture the effects of cavities and canopies on target signatures using traditional point scatter

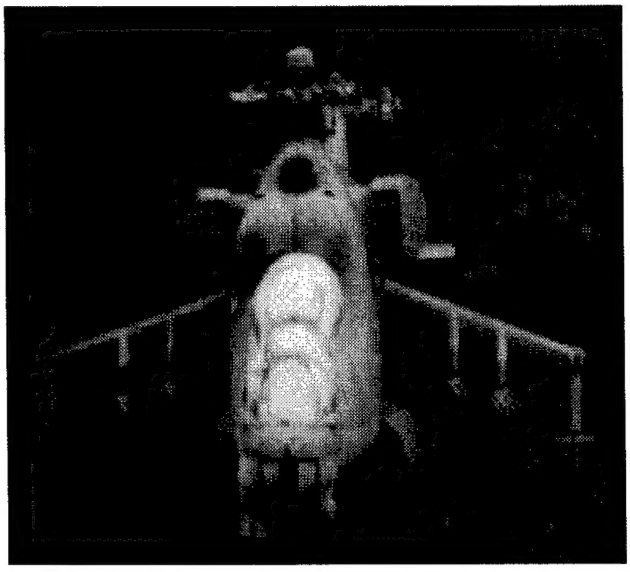


Figure 1. Photograph of Hind-D helicopter.

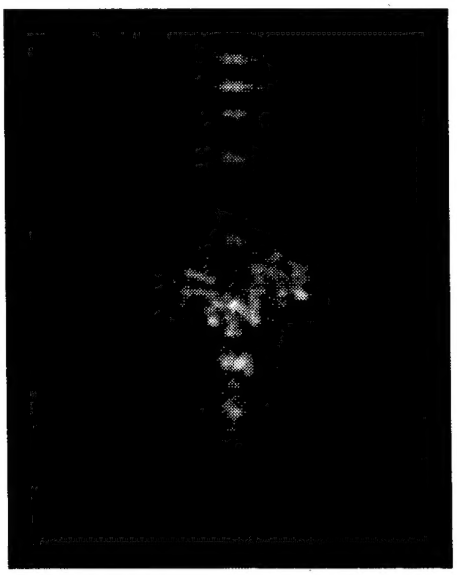


Figure 2. ISAR image of Hind-D helicopter.

modeling techniques. A simple improvement to point scatter models is the addition of frequency- and time-dependent terms to selected scatterers [5,6]. If this approach is used, traditional methods of generating point scatterers are not valid. New techniques are needed to account for the nonlinear variables.

The goal of this report is to evaluate the feasibility of developing a hybrid genetic algorithm/expert system (HGAES) to generate enhanced point scatter models from 3-D ISAR images. The methodology is to develop, implement, and analyze a simple 2-D algorithm, then extrapolate the results to determine the feasibility of developing a more complicated 3-D algorithm.

The HGAES combines ideas from several researchers. Several authors have developed genetic algorithms (GAs) to determine scattering centers from measured data [7,8]. Techniques have also been developed for combining expert systems and genetic algorithms [9,10]. A brief overview of GAs and expert systems follows.

Review

GAs can be used to solve global optimization problems. Typically, they search for the maximum value of a function using techniques that mimic natural evolution [11,12]. GAs update a population of solutions, rather than a single solution. For example, to find the maximum of a function with 10 parameters, the population would consist of several possible solutions that each contain 10 parameters. Conceptually, each parameter is considered to be a chromosome. The chromosomes in each population are randomly mutated and mixed. Chromosomes with real encoding are approximated by floating point data structures on a computer. Mutation for chromosomes with real encoding is performed by adding random values to the parameters represented by the chromosomes. Crossover or mixing occurs between chromosomes in different populations. Crossover is usually based on criteria modeled after a survival-of-the-fittest scenario. Populations that are closer to the global maximum are considered to be fitter, so there should be a greater probability that chromosomes from these populations persist. Often, the fittest realizations of the population are not changed. This is known as elitism.

Expert systems are a specialized branch of artificial intelligence and they are suited for solving specific problems that require reasoning with imprecise, missing, or *fuzzy* information and that are static in nature [13]. The principal components of an expert system are the knowledge base and the inference engine. The knowledge base stores information about the subject domain. A common storage method uses if/then rules. The inference engine gathers the information from the knowledge base and uses reasoning methods to find the solution. Two basic reasoning

methods are forward chaining and backward chaining. Forward chaining works from data to determine a conclusion. For example, a detective investigating a crime uses forward chaining methods to determine which suspect to investigate based on the available evidence. Backward chaining works from hypotheses and goals to determine supporting evidence. For example, a detective may make a hypothesis about a suspect, then collect evidence to support or refute it. Uncertainty in reasoning can be represented using fuzzy logic, Bayesian statistics, or ad hoc methods.

There are many advantages to combining a GA and an expert system. Both techniques do not require calculating the inverse of an analytic model or a gradient. This is a big advantage for modeling radar signatures because these calculations can be conceptually and computationally complex. Also, if models are generated from monopulse radar data, there is an ambiguity in the height information. This ambiguity makes it more difficult to resolve scattering centers using analytic-based techniques. A disadvantage of GAs is that they are slow. Using an expert system to guide the mutation and crossover operations in a GA should considerably reduce the amount of computer processing time.

Many aspects of modeling target signatures with point scattering centers correspond to an HGAES approach. For example, a GA could be used to adjust the position and RCS of scattering centers using mutation, and adding and deleting scattering centers could be implemented using crossover. An expert system is also suited for several aspects of modeling such signatures. It is possible for an RCS analyst to determine which pixels in an ISAR image are due to specular reflection, edge effects, and cavity resonance based on experience and knowledge of the target. Many simple rules can be determined for guiding mutation and crossover in a GA. For example, one would not mutate the position from a scattering center to a location where there is no radar return. Mutations could be guided by an expert system to resolve ambiguous height information provided by a monopulse radar image. For example, scattering centers could be split but constrained, so that their average height is unchanged.

Algorithm

In this report, an HGAES will be described and implemented that can determine an enhanced point scatter model representation of a target from 2-D ISAR images of a simple target. Time- and frequency-dependent parameters will be used to supplement a point scatter model. The algorithm will be tested using simulated radar data generated with a target model that matches the model used in the algorithm.

Far-field radar data will be simulated using a simple target and radar model. The simulated radar will be based on an existing U.S. Army Research Laboratory (ARL) *W*-band instrumentation radar [14,15]. This is

a frequency-stepped system with a total bandwidth of 640 MHz. The simulated target will consist of six isotropic point scatterers with user-selected RCS values and 2-D locations. Two of these scatterers will have frequency- and time-dependent parameters that are modeled with third-order polynomials, as shown in equation (2). The simulated electric fields will be calculated based on the principle of superposition using

$$E(k,t) = \sum_I p_i(t,k) e^{jkd_i(t)} , \tag{1}$$

$$p_i(t,k) = \sigma_i^{1/2} e^{ja_1(k-a_2)^3} e^{jb_1(t-b_2)^3} , \tag{2}$$

and

$$d_i(t) = \left| \vec{R}_0 + \vec{r}_i(t) \right| , \tag{3}$$

where $E(k,t)$ = electric field computed without any range attenuation, k = propagation number, t = time, $p_i(t,k)$ = time- and frequency-dependent backscatter coefficient of the ith scatterer, σ_i = RCS of the ith scatterer, j = complex number, b and a are time- and frequency-dependent coefficients, $d_i(t)$ = distance from the radar to the ith scatterer, R_0 = range of the radar to the center of rotation of the target, and $r_i(t)$ = 2-D location of the ith scatter relative to the target at t.

Figure 3 shows an example of a 2-D ISAR image generated from data simulated using equations (1), (2), and (3). The data consisted of simulated electric field values in meters for 64 frequencies and 64 aspect angles. Standard range-Doppler processing techniques were applied to radar data that were prewarped to lie on a rectangular grid in the spatial-frequency domain [16,17]. Normally, ISAR radar data are collected in a polar coordinate system, then interpolated to a rectangular coordinate system. This step was bypassed. The total number of scatterers simulated was six, and two of these scatterers had time- and frequency-dependent parameters. The scatterers with these parameters are located at the x,y coordinates (–2.0,–3.2) and (1.0,1.0). A constant RCS of –15 dBsm was selected for each scattering center. For standard point scatterers, the a and b coefficients in equation (2) were set to zero. For scatterers with time- and frequency-dependent parameters, the a and b coefficients were set to one in the simulation. The parameters being estimated by the HGAES are the position and RCS of the six scattering centers and the time and frequency coefficients for two scattering centers. I assume that I know the total number of scattering centers and the number of time- and frequency-dependent scattering centers.

Note that the brightness of the scattering centers in figure 3 appears to vary even though their RCS is constant. This effect is caused by the point spread function being sampled at different locations for four scattering

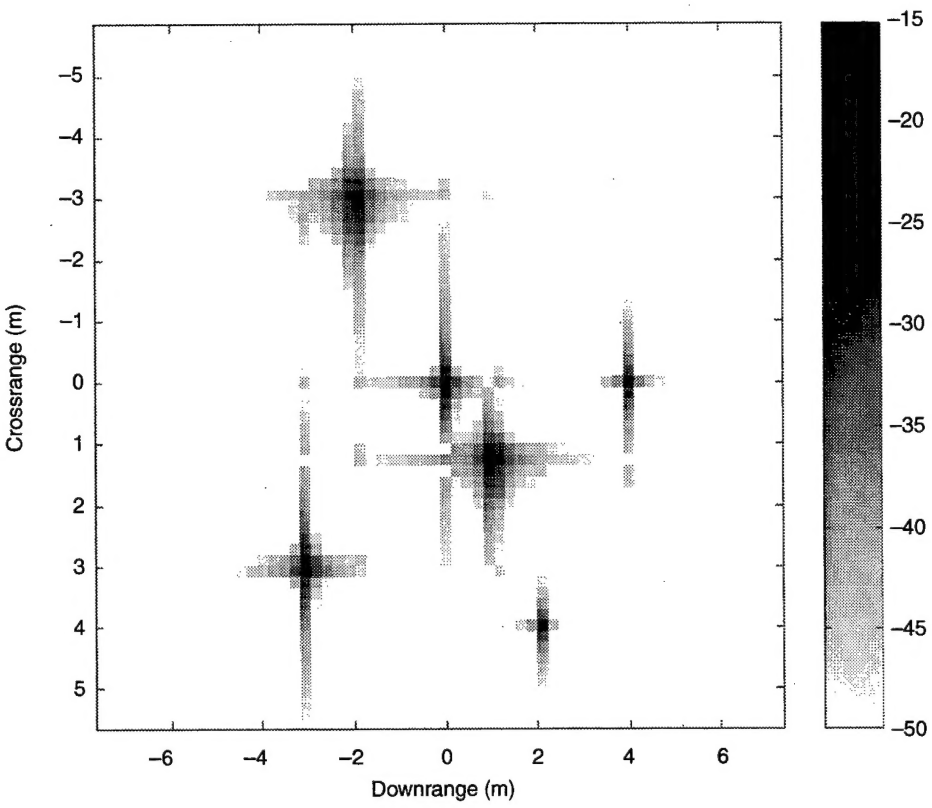


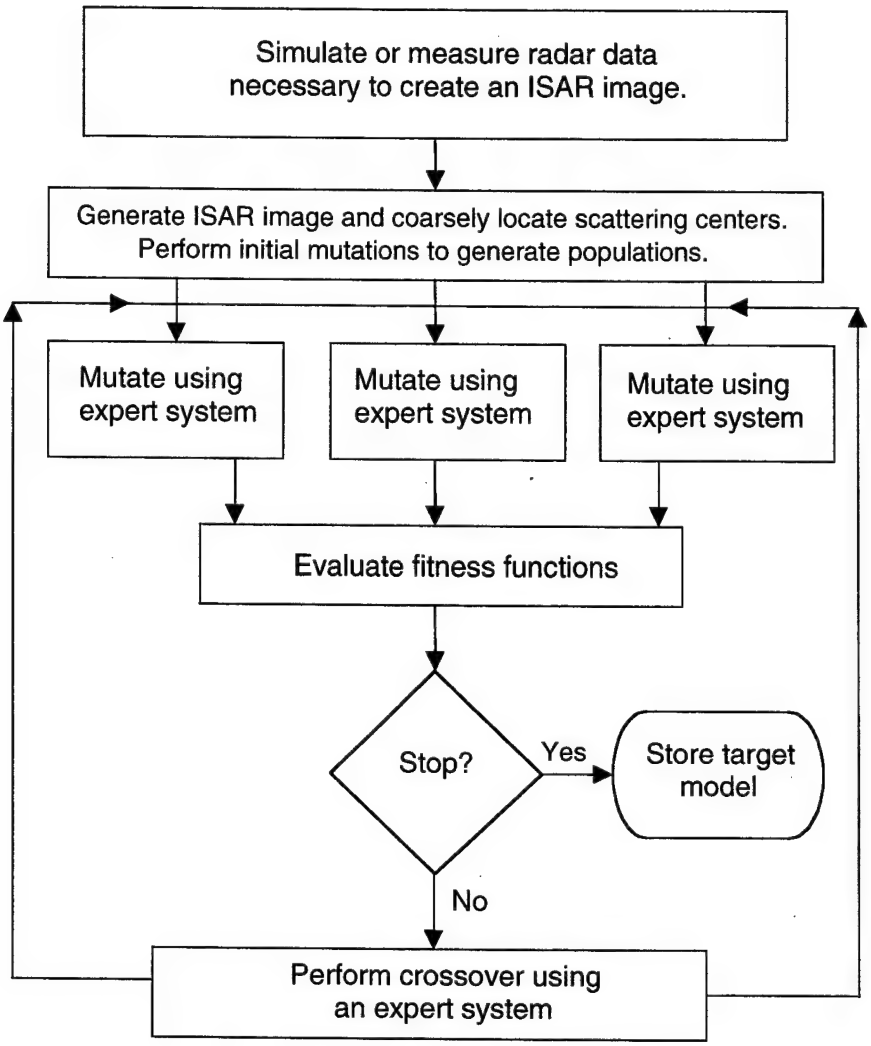
Figure 3. Example of a simulated 2-D ISAR image.

centers and by the addition of time- and frequency-dependent parameters for two scattering centers.

Figure 4 shows an overview of an HGAES used to determine an enhanced point scatter model representation of a target from measured radar data. First, simulated data will be generated and imaged using the procedure described above. Next, initial scattering centers will be determined using an ISAR peak-detection algorithm [18]. Then these scattering centers will be randomly mutated to generate additional populations. Next, the scattering centers will be mutated according to the rules described by an expert system. Then a fitness function is evaluated for each population of scattering centers. The stopping condition is currently based on the iteration number, but it could be easily changed to a fitness threshold level. Next, crossover between the populations is performed according to rules described by another expert system.

The scattering center parameters will be randomly mutated and mixed, but constrained and directed by an expert system. The knowledge base in the expert system will be represented by if/then rules generated by the author. Since the algorithm is being tested with simulated data, complete knowledge of the target is available to develop the expert system. ARL has a large database of target signatures that can be used to help develop a future knowledge base for an expert system. Inferences will be reached using forward chaining, and uncertainty will be represented using an

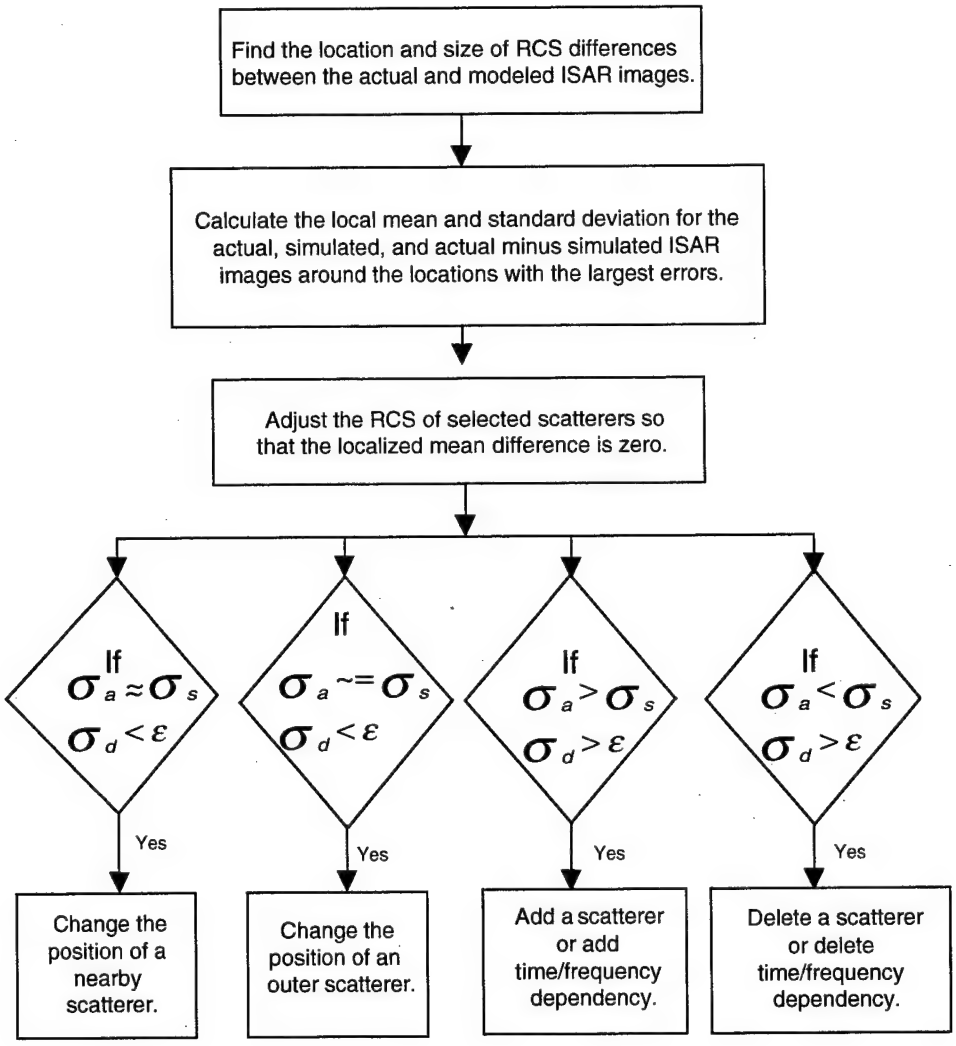
Figure 4. Flow chart of an HGAES.



ad hoc approach with both parametric and nonparametric statistics. Presently, there is no explanatory interface to help analyze conclusions. The following text describes the individual components in more detail.

Figure 5 is an outline of a simple expert system that is used to guide mutation. First, localized areas of interest are determined by examining the differences between the actual and simulated images. Then simple statistics around this location are computed in square meters. The mean and standard deviation are calculated for the actual, modeled, and actual minus modeled ISAR images. The size and order of these statistics are used to determine which scatterers to mutate. Next, the RCS of regions with larger errors are selected and adjusted so that the mean difference is zero. In the current implementation, scatterers may be created and destroyed, but the net gain or loss is constrained to be zero. Also, only one scatterer can be created or destroyed per algorithm iteration. The difference between moving versus destroying and creating a scattering center

Figure 5. Expert system used to guide mutations.



is that a scatterer is moved around a region while a scatterer can be created and destroyed from different regions.

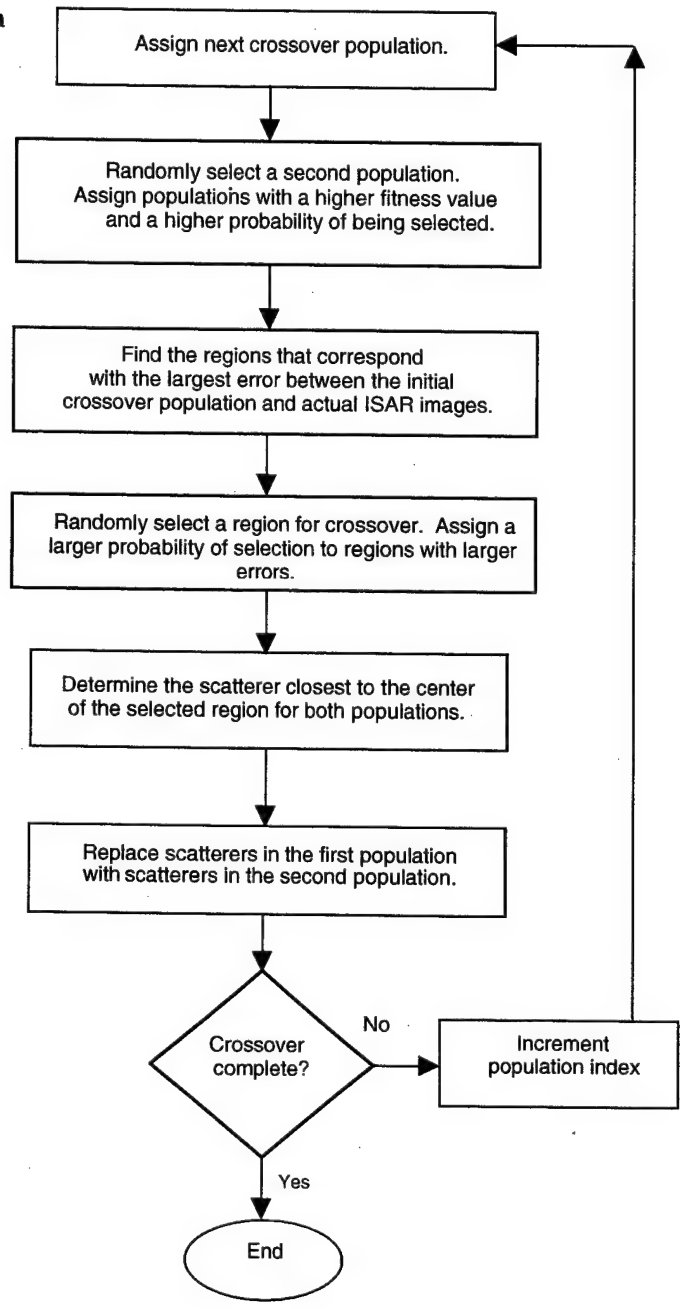
The next step in the flow chart in figure 4 is to evaluate the fitness function for the different populations of scattering centers. The fitness function selected is

$$F = -\sum_{R^2} \left(\left| g_r g_r^* - \hat{g}_r \hat{g}_r^* \right| \right) , \tag{4}$$

where R^2 = a 2-D grid sampled at the center of the pixel locations of the 2-D ISAR image, $g_r\, g_r^*$ = the simulated scatterer brightness at the rth pixel, and * = complex conjugate. The scatterer brightness is equal to the scatterer RCS plus imaging errors [19]. The absolute difference of the error was chosen rather than the squared difference of the error because of the potential large dynamic range of the radar returns.

The last step in the flow chart in figure 4 is to perform crossover between the populations of scattering centers. Figure 6 shows an overview of expert system rules used to guide crossover. First an initial population is sorted based on fitness value, then chosen based on a loop index. A second population is randomly selected, but with a bias toward populations with a higher fitness rank. Scatterer centers in the second population are not changed; they are only used to modify the first population. Next, the location and magnitude of the three largest errors between the image generated using the first population and the actual image are computed

Figure 6. Expert system used to guide crossover.



using a peak detection algorithm. A single location is randomly selected from these three choices with a higher probability assigned to locations with larger errors. Next, the closest scatterers to this location in the first population are replaced with the closest scatterers to this location in the second population. This process is repeated for each population that is not designated as elite. Keeping elite populations also results in the elimination of the populations with the lowest fitness values.

Results

The algorithm was run using six scattering centers, two of which had time and frequency dependencies. The population size was 20, the number of elite populations was 3, and 20 iterations of the algorithm were performed. A single run required approximately 1 hr of processing time. Figures 7 and 8 show the actual image and the image resulting from the HGAES algorithm. Figure 9 shows the maximum of the fitness function evaluated for the populations after each iteration. The images in figures 7 and 8 look similar, and figure 9 indicates that the algorithm converges to a stable value after approximately 13 iterations. These qualitative results indicate that the algorithm is working reasonably well.

The algorithm was coded in MATLAB® 5.3. This environment was selected because it has a short learning curve and good visualization and mathematical programming tools. The major disadvantages were its slow speed and its inability to represent knowledge with flexible data structures and objects. The total runtime was approximately 2 hr on a 333-MHz Pentium II computer.

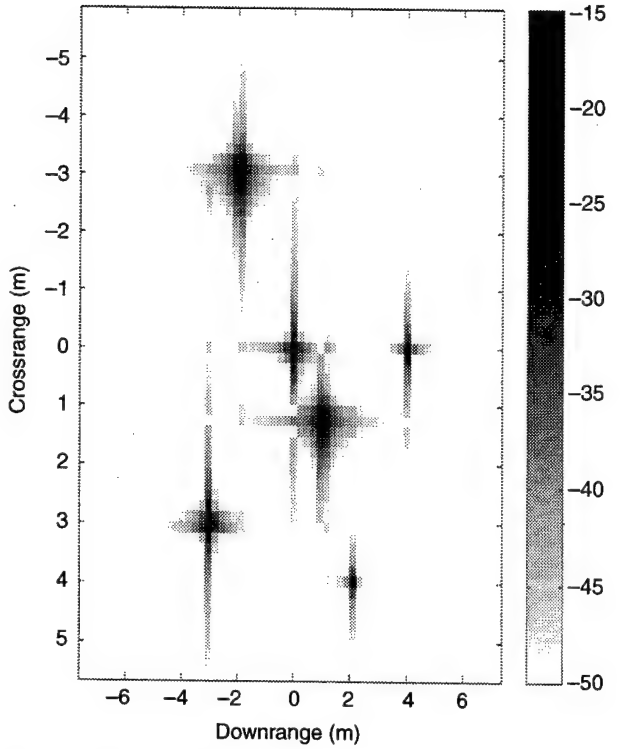


Figure 7. Actual ISAR image of 6 point scatterers, 2 with time and frequency dependencies.

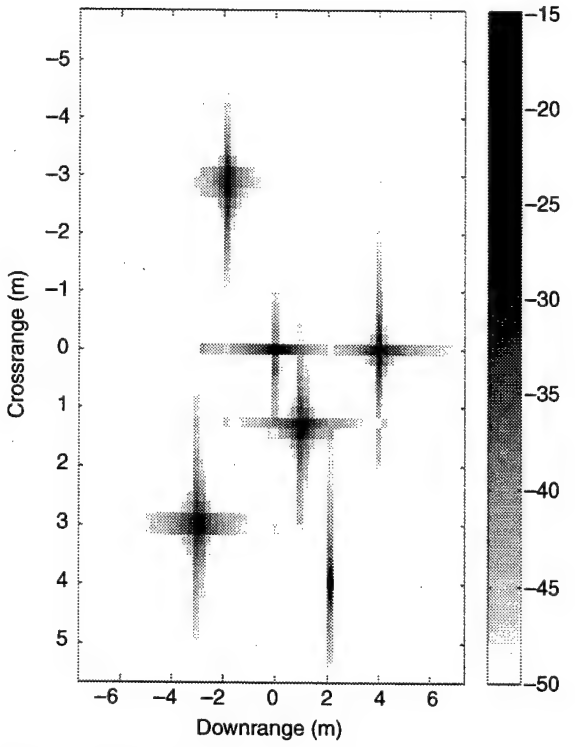


Figure 8. Image from scatterers estimated using HGAES algorithm.

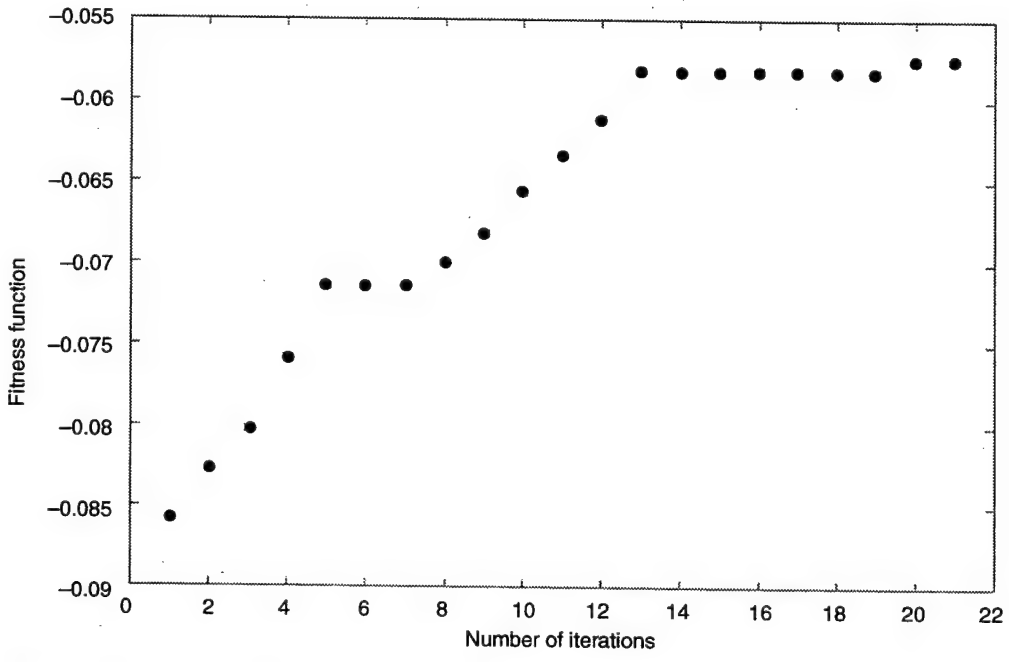


Figure 9. Realization of fitness function.

Conclusion

An HGAES was developed for modeling targets with nonlinear time- and frequency-dependent radar returns. The algorithm determined an enhanced point scatter model representation of the target from 2-D ISAR images. The model for the nonlinear parameters was relatively simple, so it should be suitable for incorporation into HWIL and SIL simulations that use point scatterer models. The algorithm was demonstrated to work on simulated radar data using a simple technique for determining the initial scattering centers.

Many of the ideas used in developing the algorithm can be extended to more complex targets and from 2-D to 3-D images. The idea of using both parametric and nonparametric statistics in the expert system was successful. Nonparametric statistics simplified the coding, while the parametric statistics were better for quantifying errors. Several lessons were learned while coding the algorithm that should apply to future algorithms. The expert system should not be developed independent of the GA and the source code. The expert system can be simplified by taking advantage of the random nature of the GA and the programming environment.

A major issue in the future development of an HGAES is knowledge representation. Efforts have been made to develop a framework for combining intelligent systems, but flexibility and speed are reduced and there is a significant learning curve. Expert system shells are available for representing knowledge, but they also lack flexibility and speed. A reasonable alternative is to develop the algorithm in an object-oriented programming language such as C++. This approach would provide speed and flexibility, but would require constructing objects. Also, future work should use more sophisticated superresolution techniques for estimating the initial position and RCS of scattering centers without time- and frequency-dependent components.

An HGAES is a powerful technique for solving nonlinear global search problems with faster convergence than a simple GA. The big advantage of this technique is that it does not require calculating the inverse of an analytic model. The downside of the approach is that it is more difficult to implement than a GA. Also, the expert system needs to be constructed so that there is sufficient variation in the mutation and crossover process to ensure convergence to a global maximum. My conclusions are that models developed using this technique have the potential to enhance the accuracy of weapon systems simulations and further investigation and development of this technique are warranted.

Acknowledgments

I would like to thank Jim Spall for introducing me to genetic algorithms and Dave Hillis for his helpful suggestions regarding the development of the technique.

References

1. R. W. Hardin, *HWIL: Technology Advances Add Reality to Missile Tests*, OE Reports, www.spie.org/web/oer/march/mar99/hwil.html (April 1999).
2. A. V. Saylor and K. R. Harrison, *Validation of Monopulse High Range Resolution Polarimetric Target Models*, U.S. Army Missile Command, TN-131-070, (October 1990).
3. A. V. Saylor and N. Evers, *Point Scatterer Model Development and Validation for 95 GHz T-62*, U.S. Army Missile Command, TN-131-108 (January 1992).
4. W. P. Yu, L. D. To, and K. Oh, "N-Point Scatterer Model RCS/Glint Reconstruction from High-Resolution ISAR Target Imaging," *10th Annual ITEA Symposium*, Session IV (1991), pp 67–75.
5. A. W. Rihaczek and S. J. Hershkowitz, "Man-Made Target Backscatter Behavior of Conventional Radar Resolution Theory," *IEEE Trans. Aerosp. Electron. Syst.*, **32**, 2 (April 1996), pp 809–824.
6. A. W. Rihaczek and S. J. Hershkowitz, *Radar Resolution and Complex-Image Analysis*, Artech House (1996).
7. E. J. Hughes and M. Leyland, "A Multi-Species Genetic Algorithm Applied to Radar Scattering Centre Identification in Three-Dimensions," *Genetic Algorithms in Engineering Systems: Innovations and Applications*, No. 446 (September 1997), pp 472–477.
8. Q. Li, E. J. Rothwell, K-M. Chen, and D. P. Nyquist, "Scattering Center Analysis of Radar Targets Using Fitting Scheme and Genetic Algorithm," *IEEE Trans. Antennas Propag.*, **44**, 2 (February 1996), pp 198–207.
9. P. V. Rocha, S. K. Khebbal, and P. C.Treleaven, "A Framework for Hybrid Intelligent Systems," *Proceedings of the First New Zealand International Two-Stream Conference on Artificial Neural Networks and Expert Systems* (1993), pp 206–209.
10. Y. H. Song and A. T. Johns, "Application of Fuzzy Logic in Power Systems: Part 2: Comparison and Integration with Expert Systems, Neural Networks, and Genetic Algorithms," *Power Eng. J.* (August 1998), pp 185–190.
11. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MI (1975).

12. D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, MA (1989).

13. R. Forsyth, "Artificial Intelligence: Principles and Applications," *The Anatomy of Expert Systems*, Chapman and Hall Computing (1986), pp 186–199.

14. S. R. Stratton, H. B. Wallace, and R. L. Bender, *Radar Cross-Section Measurements of the ZSU-23-4 at X-, K_a- and W-bands*, U.S. Army Research Laboratory, ARL-TR-1722 (September 1998).

15. R. L. Bender, *Use of a Remotely Controlled Dihedral for Calibrating a Polarimetric Radar*, U.S. Army Research Laboratory, ARL-MR-318 (June 1996).

16. D. R. Wehner, *High Resolution Radar*, Artech House, Inc. (1987).

17. W. G. Carrara, R. S. Goodman, and R. M. Majewski, *Spotlight Synthetic Aperture Radar*, Artech House (1995), pp 157–201.

18. G. H. Goldman and H. Dropkin, *High Frequency Iterative Autofocus Algorithm for Noncooperative ISAR*, U.S. Army Research Laboratory, ARL-TR-1890 (May 1999).

19. J. P. Skinner, B. M. Kent, R. C. Wittmann, D. L. Mensa, and D. J. Andersh, "Normalization and Interpretation of Radar Images," *IEEE Trans. Antennas Propag.*, **46**, 4 (April 1998), pp 502–506.

Appendix.—Main Function for HGAES Code

```
% genetic algorithm/expert system that estimate time and frequency dependent scattering
% centers from simulated RF data radar simulates 2-D ISAR data collected with a
% rectangular frequency collection space.
% Written by Jeff Goldman, June 1999
%
function hgaes(seed)

global c
c=3e8;

randn('seed',seed);
rand('seed',seed);

load 'scat_ft.dat';  % correct model in file
[num_scat,nine]=size(scat_ft);  % use correct number of scatterers

carrier = 95e9;    % radar simulation parameters
freq_step = 10e6;  % fx and fy are exactly on rectangular grid in simulation
num_az = 64;
num_freq = 64;
ang_inc = (0.5*pi/180)/num_az;

drange_len_far = c/(2*freq_step);     % unambiguous down range
crange_len_far = c/((carrier+freq_step*(num_freq-1)/2)*2*ang_inc); % cross range

TF_INDEX=5;
num_scat_tf=sum(scat_ft(:,TF_INDEX));
isar_truth_m2=abs(gen_isar_ft(carrier,freq_step,num_az,num_freq,ang_inc,scat_ft)).^2; %
generate image

fir2coef=ones(3,3)*0.05;  % 2-D filter for smoothing ISAR image
fir2coef(2,2)=.55;

isar_truth_filt_m2=filter2(fir2coef,isar_truth_m2);
RCS_truth=sum(sum(isar_truth_filt_m2));

if (1)  % define image parameters
  min_im= -50;
  max_im= -15;
  dr_start=1;
  dr_end=num_freq - dr_start  + 1;
  cr_start=1;
  cr_end=num_az - cr_start + 1;
  x_axis=((dr_start:dr_end)-num_freq/2 -1)*drange_len_far/(num_freq);
  y_axis=((cr_start:cr_end)-num_az/2 -1.)*crange_len_far/(num_az);
end

if (1)  % plot original image
  image_dBsm = 10.*log10(isar_truth_m2)';
  figure
  imagesc(x_axis,y_axis,image_dBsm,[min_im max_im])
  colorbar('vert')
  title('ISAR image far-field, exact, with time/frequency dependency')
  xlabel('down range (m)')
  ylabel('cross range (m)')
end

NPOP=20;          % population size
n_cases=20;        % number of iterations
N_elete=3;        % number of elete
P_mutate=0.35;    % probability of mutation for scatterer
P_del_tf_scat=.4; % probabilty of deleting a tf comp of a scatterer, (or 1-P of creating)
P_add_del=0.2;    % probability of adding or deleting a scatterer
```

```
RN_mut_scat_ele_array=[0.1 0.1 0.0 0.1]; % real number associated with mutation proabability
for SC
%  [x,y,z,amp,time/freq, add/del scat]  tf=binary flag
RN_mut_tf_ele_array= [0.05 0.05]; % real number associated with mutation proabability for
time/freq variation
                                    % mutate time and frequency simulataneously
%[freq amp,freq offset,time amp, time offset];

SCAT_ELE=5;  % number of elements for a standard scattering center
TF_SD_ELE=4;  % number of elements for the time and frequency dependent scattering
center
N_MIN_SCAT_MUT=1;  % minumum number of scatterers to mutate

% used only during initialization
P_mut_scat_ele_array=cumsum(RN_mut_scat_ele_array)/sum(RN_mut_scat_ele_array);  %
normalized cumulative mutation probability
P_mut_tf_ele_array=cumsum(RN_mut_tf_ele_array)/sum(RN_mut_tf_ele_array);  %
normalized cumulative mutation probability

SD_mut_xyz=0.1*[1 1 1]; % in meters
perc_mut_amp=[0.1];  % amp in meters
SD_mut_tf=[1 1 1 1]; %units = [none,rad,none,rad];

ndr_cells_rcs=7;  % make odd
ncr_cells_rcs=7;  % make odd
num_nn=3;   % number of nearest neighbors in region of interest
num_nn1=1;  % number of nearest neighbor

INIT_MIN_MUT=2;  % initial minimum mutation for initial population (elite)

% generate population of solutions

fitness_array=zeros(1,NPOP);
scat_ft_models=zeros(num_scat,SCAT_ELE+TF_SD_ELE,NPOP);  % GA population
temp_ft_models=zeros(num_scat,SCAT_ELE+TF_SD_ELE,NPOP);  % temporary storage
loss_array_mut=zeros(1,NPOP);
loss_array_cross=zeros(1,NPOP);  % crossover

init_process_variable  % initialize structure process, required by find_peaks
process.num_scat=num_scat;

[idr_array,icr_array,w_array,rcs_array]=find_peaks(isar_truth_m2,process) % sorted by best
match to template
x_init=((idr_array)-num_freq/2 -1)*drange_len_far/(num_freq); % x positions of peaks
y_init=((icr_array)-num_az/2 -1.)*crange_len_far/(num_az);    % y positions of peaks
scat_ft_models(:,1,1)=x_init;
scat_ft_models(:,2,1)=y_init;
scat_ft_models(:,4,1)=rcs_array.^0.5;  % amplitude

n_mut_array=sum(rand(num_scat-INIT_MIN_MUT,NPOP) < P_mutate) + INIT_MIN_MUT;
% number of scatterers to mutate
for ipop=1:NPOP  % mutate initial estimate to form population
   scat_ft_models(:,:,ipop)=scat_ft_models(:,:,1);     % initialize to best guess
end

for ipop=1:NPOP  % mutate initial estimate to form population
   imut_scat_array=ceil(num_scat*rand(1,n_mut_array(ipop)));  % select scattering centers to
mutate
   while (imut_scat_array(1)==imut_scat_array(2)) % make sure first two elements are not
equal
                                                  % fix later to be more general
      imut_scat_array=ceil(num_scat*rand(1,n_mut_array(ipop)));
   end
   for itf=1:num_scat_tf
      iscat=imut_scat_array(itf);
      scat_ft_models(iscat,TF_INDEX,ipop)=1;
      scat_ft_models(iscat,6:9,ipop)=SD_mut_tf.*rand(1,4) + 0.5*ones(1,4);
```

```
   end
   for iscat=1:n_mut_array(ipop)  % loop through scatterers for the ipop population
      iscat_ele=get_rand_index(P_mut_scat_ele_array);  % find scattering element
      if (iscat_ele <= 3) % add gaussian pertubation to position of scatterer
            scat_ft_models(iscat,iscat_ele,ipop)=scat_ft_models(iscat,iscat_ele,ipop) + ...
            SD_mut_xyz(iscat_ele)*randn(1);
      elseif (iscat_ele==4) % for RCS , multiple by gaussian pertubation
            scat_ft_models(iscat,iscat_ele,ipop)=scat_ft_models(iscat,iscat_ele,ipop)* ...
            (1 + perc_mut_amp*randn(1));
      else
         error('iscat should be 4 or less')
      end
   end
end

original=scat_ft_models(:,:,1)
mut_image=scat_ft_models(:,:,2)

if (0)  % plot best estimate
  isar_mut=gen_isar_ft(carrier,freq_step,num_az,num_freq,ang_inc,scat_ft_models(:,:,1)); %
generate image
  image_m2=(abs(isar_mut(dr_start:dr_end,cr_start:cr_end))).^2;
  image_dBsm = 10.*log10(image_m2)';
  figure
  imagesc(x_axis,y_axis,image_dBsm,[min_im max_im])
  colorbar('vert')
  title('peak ISAR image far-field, exact, time/frequency dependent')
  xlabel('down range (m)')
  ylabel('cross range (m)')
end

% start GA
% first mutate all scatterers, no elete, than crossover

if (N_MIN_SCAT_MUT < 1)
    error('N_MIN_SCAT_MUT >= 1')  % mutate atleast one scattering center
end

for icases=1:n_cases
 for ipop=1:NPOP % rerank after crossover
  isar_sim=gen_isar_ft(carrier,freq_step,num_az,num_freq,ang_inc,scat_ft_models(:,:,ipop));
% generate image
  isar_sim_filt_m2=filter2(fir2coef,abs(isar_sim).^2);
  isar_diff=(isar_truth_filt_m2-isar_sim_filt_m2);
  loss_array_cross(ipop)=sum(sum(abs(isar_diff)));
 end
 [temp,ifitness_cross]=sort(loss_array_cross);
 loss_min_cross=temp(1);
 loss_array_mut(ifitness_cross(1))=loss_array_cross(ifitness_cross(1));  % propagate the elite
population

 for iipop=2:NPOP  % 1 elete for mutation
   ipop=ifitness_cross(iipop);
   ipop_in_mut=ipop;
   scat_mut_array='';   % variable size, start over for each population
   region_mut_array='';
   n_scat_mut=sum(rand(1,num_scat) < P_mutate);
   if (n_scat_mut < N_MIN_SCAT_MUT)
      n_scat_mut=N_MIN_SCAT_MUT;
   end

   isar_sim=gen_isar_ft(carrier,freq_step,num_az,num_freq,ang_inc,scat_ft_models(:,:,ipop));
% generate image
   isar_sim_filt_m2=filter2(fir2coef,abs(isar_sim).^2);
   isar_diff=abs(isar_truth_filt_m2-isar_sim_filt_m2);
   isar_abs_diff=abs(isar_diff);
```

```
   process.num_scat=n_scat_mut;
   process.templete=1;
   process.n_remove_dr=7; % make odd
   process.n_remove_cr=7; % make odd
   [idr_array,icr_array,w_array,rcs_array]=find_peaks(isar_abs_diff,process);   % sorted by best
match to template
   x_array=((idr_array)-num_freq/2 -1)*drange_len_far/(num_freq);
   y_array=((icr_array)-num_az/2 -1.)*crange_len_far/(num_az);

   nmean_diff_array='';
   mean_diff_array='';
   nstd_diff_array='';

   % generate statistics to guide mutation
   for imut=1:n_scat_mut  % calculate statistics, mutate regions with largest errors first

[rcs_mean_diff,rcs_std_diff]=local_rcs_stat(isar_diff,idr_array(imut),icr_array(imut),ndr_cells_
rcs,ncr_cells_rcs);
     [rcs_sim_mean,rcs_sim_std]=
local_rcs_stat(isar_sim_filt_m2,idr_array(imut),icr_array(imut),ndr_cells_rcs,ncr_cells_rcs);

[rcs_true_mean,rcs_true_std]=local_rcs_stat(isar_truth_filt_m2,idr_array(imut),icr_array(imut),
ndr_cells_rcs,ncr_cells_rcs);
     % rcs_mean_ratio=rcs_sim_mean/rcs_true_mean;
     std_ratio_array(imut)=(rcs_true_std -rcs_sim_std)/(rcs_sim_std + rcs_true_std);
     nmean_diff_array(imut)=rcs_mean_diff/(rcs_sim_mean+rcs_true_mean);
     mean_diff_array(imut)=rcs_mean_diff;
     nstd_diff_array(imut)=rcs_std_diff/(rcs_sim_std+rcs_true_std);
  end

  % first determine tf scatterers to add/delete
  del_tf_flag=(rand(1) < P_del_tf_scat);  % add/delete only one scatterer per mutation
  del_scat_flag= (rand(1) < P_add_del);
  n_scat_move = n_scat_mut -del_tf_flag -del_scat_flag;
  iscat_mut=0; % counter to keep track of scatterers mutated

  if (del_tf_flag)
       score_move_array='';
       score_add_tf_array='';
       del_tf_scat = num_scat + 1 -ceil(rand(1)*num_scat_tf);   % randomly determined which
tf scatterer to delete
       % fix later, make smarter, not random
       [temp,itf_array]=sort(scat_ft_models(:,TF_INDEX,ipop));    % itf last elements
       [ireg,temp] = find_region(scat_ft_models(itf_array(del_tf_scat),1,ipop),...
                     scat_ft_models(itf_array(del_tf_scat),2,ipop),x_array,y_array);

       scat_ft_models(itf_array(del_tf_scat),TF_INDEX:9,ipop)=zeros(1,5);

       iscat_mut=iscat_mut + 1;
       scat_mut_array(iscat_mut)=itf_array(del_tf_scat);
       region_mut_array(iscat_mut)=ireg;

       [temp,instd_diff_array]=sort(nstd_diff_array);  % determine which scatterer to add
       [temp,istd_ratio_array]=sort(std_ratio_array);
       for iregion=1:n_scat_mut  % large score better, limit search to number of mutations
           score_add_tf_array(iregion)=find(instd_diff_array==iregion) +
find(istd_ratio_array==iregion);
       end

       [temp,iadd_tf]=sort(score_add_tf_array); % large score indicates add scatterer
       % fix, check to make sure not adding and deleting same tf scatterer


[inn1,temp]=find_nn(x_array(iadd_tf(n_scat_mut)),y_array(iadd_tf(n_scat_mut)),num_nn1,sca
t_ft_models(:,:,ipop));
       ireg=iadd_tf(n_scat_mut);
       loop=0;
```

```
      while (scat_ft_models(inn1(1),TF_INDEX,ipop)==1  & n_scat_mut - loop -1 > 0)  %if
already set, find another scatterer
         loop=loop + 1;
         [inn1,temp]=find_nn(x_array(iadd_tf(n_scat_mut-loop)),y_array(iadd_tf(n_scat_mut-
loop)),num_nn1,scat_ft_models(:,:,ipop));
         ireg=iadd_tf(n_scat_mut-loop);
      end
      if (loop==num_scat_tf)  % if stuck on scatterer, force a change
         inn_rnd=ceil(rand(1)*(num_scat_tf-1));
         if (inn_rnd==inn1)
            inn1=inn1 + 1;
         else
            inn1=inn_rnd;
         end
         [ireg,temp] =
find_region(scat_ft_models(inn1,1,ipop),scat_ft_models(inn1,2,ipop),x_array,y_array);
      end

      % use nearest neighbor
      scat_ft_models(inn1(1),TF_INDEX,ipop)=1;
      scat_ft_models(inn1,6:9,ipop)=0.5*ones(1,4) + randn(1,4);
      iscat_mut=iscat_mut + 1;
      scat_mut_array(iscat_mut)=inn1;
      region_mut_array(iscat_mut)=ireg;
   end
   if (del_scat_flag)
      score_add_tf_array='';
      score_move_array='';
      tf_scat_flag=1;
      while (tf_scat_flag) % fix later, make smarter, not random
        del_scat = ceil(rand(1)*num_scat);   % randomly determined which scatterer to delete
        tf_scat_flag=scat_ft_models(del_scat,5,ipop);
      end

      [ireg,temp] = find_region(scat_ft_models(del_scat,1,ipop),...
                        scat_ft_models(del_scat,2,ipop),x_array,y_array);

      iscat_mut=iscat_mut + 1;
      scat_mut_array(iscat_mut)=del_scat;
      region_mut_array(iscat_mut)=ireg;

      [temp,instd_diff_array]=sort(nstd_diff_array);  % determine which scatterer to add
      [temp,istd_ratio_array]=sort(std_ratio_array);
      for iregion=1:n_scat_mut  % large score better, limit search to number of mutations
          score_add_tf_array(iregion)=find(instd_diff_array==iregion) +
find(istd_ratio_array==iregion);
      end
      [temp,iadd_tf]=sort(score_add_tf_array); % large score indicates add scatterer
      % fix, check to make sure not adding and deleting same tf scatterer


[inn1,temp]=find_nn(x_array(iadd_tf(n_scat_mut)),y_array(iadd_tf(n_scat_mut)),num_nn1,sca
t_ft_models(:,:,ipop));
      ireg=iadd_tf(n_scat_mut);
      loop=0;
      while (scat_ft_models(inn1(1),TF_INDEX,ipop)==1  & n_scat_mut-loop-1 >= 1)  %if
already set, find another scatterer
         loop=loop + 1;
         [inn1,temp]=find_nn(x_array(iadd_tf(n_scat_mut-loop)),y_array(iadd_tf(n_scat_mut-
loop)),num_nn1,scat_ft_models(:,:,ipop));
         ireg=iadd_tf(n_scat_mut-loop);
      end
      % use nearest neighbor

      scat_ft_models(inn1(1),1,ipop)=x_array(iadd_tf(n_scat_mut-loop));
      scat_ft_models(inn1(1),2,ipop)=y_array(iadd_tf(n_scat_mut-loop));
      % keep RCS the same
```

```
    end

    if (n_scat_move>=1)
       score_add_tf_array='';
       score_move_array='';
       [temp,instd_diff_array]=sort(nstd_diff_array);  % determine which scatterers to move
       [temp,istd_ratio_array]=sort(abs(std_ratio_array - mean(std_ratio_array)));
       n_scat_mut_iregion=n_scat_mut;
       for iregion=1:n_scat_mut  % large score better
           score_move_array(iregion)=find(instd_diff_array==iregion) +
find(istd_ratio_array==iregion);
       end
       [temp,imove_array]=sort(score_move_array);
       for imove=1:n_scat_move

[inn1,temp]=find_nn(x_array(imove_array(imove)),y_array(imove_array(imove)),num_nn1,sca
t_ft_models(:,:,ipop));
         iscat_mut=iscat_mut+1;  % allow for duplication in movement, no checking
         scat_mut_array(iscat_mut)=inn1;
         region_mut_array(iscat_mut)=imove_array(imove);
         iele=ceil(rand(1)*2);  % chose between x and y
         scat_ft_models(inn1(1),iele,ipop)=...
               scat_ft_models(inn1(1),iele,ipop) + rand(1)*SD_mut_xyz(iele);
       end
    end

  % adjust local RCS for each region

  for ireg=1:n_scat_mut
       if (~sum(find(region_mut_array==ireg))) % if region not mutated
          [inn1,temp]=find_nn(x_array(ireg),y_array(ireg),num_nn1,scat_ft_models(:,:,ipop));
         if (~sum(find(scat_mut_array==inn1))) % if scatter not mutated
            iscat_mut=iscat_mut + 1;
            scat_mut_array(iscat_mut)=inn1;
            region_mut_array(iscat_mut)=ireg;
            scat_ft_models(inn1,4,ipop)=...
               (abs(scat_ft_models(inn1,4,ipop)^2 + mean_diff_array(ireg) + rand(1)*0.05))^0.5;
         end
       end
  end

  if (sum(scat_ft_models(:,5,ipop)')==1)  % make sure two tf scatterers
   icheck=ceil(rand(1)*num_scat);
   while (scat_ft_models(icheck,5,ipop)==1)
       icheck=ceil(rand(1)*num_scat);
   end
   scat_ft_models(icheck,TF_INDEX,ipop)=1;
   scat_ft_models(icheck,6:9,ipop)=0.5*ones(1,4) + randn(1,4);
  end

  % evaluate fitness functions
  isar_sim=gen_isar_ft(carrier,freq_step,num_az,num_freq,ang_inc,scat_ft_models(:,:,ipop));
% generate image
  isar_sim_filt_m2=filter2(fir2coef,abs(isar_sim).^2);
  isar_diff=(isar_truth_filt_m2-isar_sim_filt_m2);
  loss_array_mut(ipop)=sum(sum(abs(isar_diff)));
end  %ipop

[temp,ifitness_mut]=sort(loss_array_mut);
loss_min_mut=loss_array_mut(ifitness_mut(1));
loss_case_array(icases)=loss_array_mut(ifitness_mut(1));

if (1)  % plot image
  intermediate_model=scat_ft_models(:,:,ifitness_mut(1));

isar_final=gen_isar_ft(carrier,freq_step,num_az,num_freq,ang_inc,scat_ft_models(:,:,ifitness_
mut(1))); % generate image
```

```
  image_m2=(abs(isar_final(dr_start:dr_end,cr_start:cr_end))).^2;
  image_dBsm = 10.*log10(image_m2)';
  figure
  imagesc(x_axis,y_axis,image_dBsm,[min_im max_im])
  colorbar('vert')
  title('Intermediate ISAR image far-field, exact, time/frequency dependent')
  xlabel('down range (m)')
  ylabel('cross range (m)')
end

temp_ft_models=scat_ft_models;
sort_models=scat_ft_models;

N_CROSS_REGIONS=3;            % used to random generate regions for crossover
n_crossover=ceil(2*rand(1)); % number of scatterers to crossover
CROSS_POWER_POP=1.5;          % weight crossover to better fitness values, larger values
better

% reorder model from best to worst, count elete twice
for ipop=1:N_elete
  scat_ft_models(:,:,ipop)=temp_ft_models(:,:,ifitness_mut(ipop));
end

for ipop=1:(NPOP-N_elete)
  scat_ft_models(:,:,ipop+N_elete)=temp_ft_models(:,:,ifitness_mut(ipop));
end
for ipop=1:NPOP
  sort_models(:,:,ipop)=temp_ft_models(:,:,ifitness_mut(ipop));
end

for ipop=1+N_elete:NPOP       % ipop is selection for crossover
   ipop_new=ceil((rand(1)^CROSS_POWER_POP)*(NPOP-1));  % weighted crossover

   if (ipop-N_elete==ipop_new)
      ipop_new=ipop_new+1;
   end

   isar_sim=gen_isar_ft(carrier,freq_step,num_az,num_freq,ang_inc,scat_ft_models(:,:,ipop));
% generate image
   isar_sim_filt_m2=filter2(fir2coef,abs(isar_sim).^2);
   isar_abs_diff1=abs((isar_truth_filt_m2-isar_sim_filt_m2));
   iroi=ceil(rand(1)*N_CROSS_REGIONS);
   process.num_scat=iroi;
   [idr_array,icr_array,w_array,rcs_array]=find_peaks(isar_abs_diff,process); % sorted by best
match to template
   x=((idr_array(iroi))-num_freq/2 -1)*drange_len_far/(num_freq);
   y=((icr_array(iroi))-num_az/2 -1.)*crange_len_far/(num_az);
   [inn_new,temp]=find_nn(x,y,n_crossover,sort_models(:,:,ipop_new));
   [inn1,temp]=find_nn(x,y,n_crossover,scat_ft_models(:,:,ipop));

   new_tf=0;
   inn1_tf=0;
   for icross=1:n_crossover  % count tf
      new_tf= new_tf + sort_models(inn_new(icross),TF_INDEX,ipop_new);
      inn1_tf=inn1_tf + scat_ft_models(inn1(icross),TF_INDEX,ipop);
   end
   if (inn1_tf==new_tf)  % maintain same number of time frequency scatterers
      for icross=1:n_crossover
          scat_ft_models(inn1(icross),:,ipop)=sort_models(inn_new(icross),:,ipop_new);
      end
   end
  end
 end % n_cases

 %model_after=scat_ft_models(:,:,ifitness_mut(1))

 for ipop=1:NPOP
```

```
  isar_sim=gen_isar_ft(carrier,freq_step,num_az,num_freq,ang_inc,scat_ft_models(:,:,ipop));
% generate image
  isar_sim_filt_m2=filter2(fir2coef,abs(isar_sim).^2);
  isar_diff=(isar_truth_filt_m2-isar_sim_filt_m2);
  loss_array(ipop)=sum(sum(abs(isar_diff)));
end

[temp,iloss]=sort(loss_array)

final_scat_model=scat_ft_models(:,:,iloss(1))
loss_case_array(n_cases+1)=loss_array(iloss(1));

if (1)  % plot final image

isar_final=gen_isar_ft(carrier,freq_step,num_az,num_freq,ang_inc,scat_ft_models(:,:,iloss(1)))
; % generate image
  image_m2=(abs(isar_final(dr_start:dr_end,cr_start:cr_end))).^2;
  image_dBsm = 10.*log10(image_m2)';
  figure
  imagesc(x_axis,y_axis,image_dBsm,[min_im max_im])
  colorbar('vert')
  title('Final ISAR image far-field, exact, time/frequency dependent')
  xlabel('down range (m)')
  ylabel('cross range (m)')
end

if (1) % plot fitness function
  figure
  plot(-loss_case_array)
  title('Final results')
  xlabel('number of iterations')
  ylabel('Fitness function')
end
```

Distribution

Admnstr
Defns Techl Info Ctr
Attn DTIC-OCP
8725 John J Kingman Rd Ste 0944
FT Belvoir VA 22060-6218

US Army Aviation Missile Cmnd
Attn AMSAM-RD-SS-HW R Applegate
Redstone Arsenal AL 35898-5000

Johns Hopkins University Applied Physics Lab
Attn J Spall
11100 Johns Hopkins Rd
Laurel MD 20723-6099

Simulation Techl
Attn J Cole
Attn S McFarlen
Attn A V Saylor
3307 Bob Wallace Ave SW 4
Huntsville AL 35805-4066

US Army Rsrch Lab
Attn AMSRL-SE-RM S Stratton
Attn AMSRL-SE-RM R Tan
Aberdeen Proving Ground MD 21005

US Army Rsrch Lab
Attn AMSRL-DD J Rocchio
Attn AMSRL-CI-LL Techl Lib (3 copies)
Attn AMSRL-CS-AS Mail & Records Mgmt
Attn AMSRL-CS-EA-TP Techl Pub (3 copies)
Attn AMSRL-SE J Mait
Attn AMSRL-SE-R A Sindoris
Attn AMSRL-SE-R B Wallace
Attn AMSRL-SE-RM C Ly
Attn AMSRL-SE-RM E Burke
Attn AMSRL-SE-RM G Goldman (25 copies)
Attn AMSRL-SE-RM H Dropkin
Attn AMSRL-SE-RM J Silverstein
Attn AMSRL-SE-RM J Silvious
Attn AMSRL-SE-RM K Tom
Attn AMSRL-SE-RM R Wellman
Attn AMSRL-SE-RM T Pizzillo
Attn AMSRL-SE-RU B Scheiner
Attn AMSRL-SE-RU R Damarla
2800 Powder Mill Rd
Adelphi MD 20783-1197

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	October 1999	Final, 1/99–6/99

4. TITLE AND SUBTITLE Preliminary Study of a Hybrid Genetic Algorithm/Expert System for Modeling Complex Radar Signatures

5. FUNDING NUMBERS
DA PR: AH16
PE: 62120A

6. AUTHOR(S) Geoffrey H. Goldman

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army Research Laboratory
Attn: AMSRL-SE-RM email: ggoldman@arl.mil
2800 Powder Mill Road
Adelphi, MD 20783-1197

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TR-2028

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. Army Research Laboratory
2800 Powder Mill Road
Adelphi, MD 20783-1197

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
ARL PR: 9NE4H1
AMS code: P622120.H16

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*
I made an initial study of a hybrid genetic algorithm/expert system (HGAES) to model targets with nonlinear radar imaging effects caused by features such as cavities and canopies. The model for the nonlinear parameters was relatively simple, so it should be suitable for incorporation into hardware-in-the-loop and software-in-the-loop simulations that currently use point scatter models. I demonstrated the algorithm on simulated two-dimensional (2-D) inverse synthetic aperture radar (ISAR) images using a simple technique to determine the initial scattering centers. Many of the ideas used in developing the algorithm can be extended to more complex targets and from 2-D to 3-D images. A major issue in the development of an HGAES is knowledge representation. My conclusions are that models determined using this technique have the potential to enhance the accuracy of weapon systems simulations; thus, this technique is worth further investigation.

14. SUBJECT TERMS
ISAR, cavities

15. NUMBER OF PAGES
29

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102